



### WHAT WE'RE DOING:

To this point we have controlled light, motion, and electrons. Let's tackle sound next. But sound is an analog phenomena, how will our digital Arduino cope?

We will once again rely on its incredible speed which will let it mimic analog behavior. To do this, we will attach a piezo element to one of the Arduino's digital pins. A piezo element makes a clicking sound each time it is pulsed with current. If we pulse it at the right frequency (for example 440 times a second to make the note middle A) these clicks will run together to produce notes. Let's get to experimenting with it and get your Arduino playing "Twinkle Twinkle Little Star".

### THE CIRCUIT:

#### Parts:



**CIRC-06**  
Breadboard Sheet  
x1



**2 Pin Header**  
x4

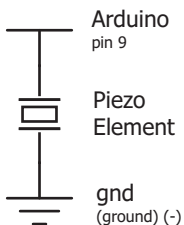


**Piezo Element**  
x1



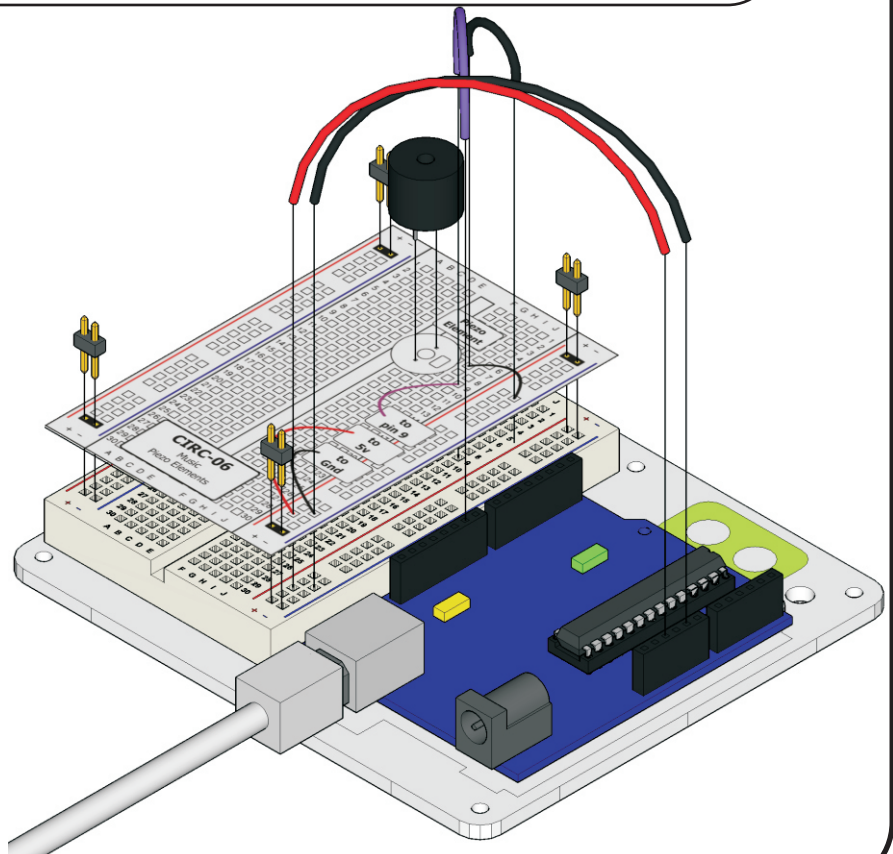
**Wire**

#### Schematic



#### The Internet

..:download:..  
breadboard layout sheet  
<http://ardx.org/BBLS06>  
..:view:..  
assembly video  
<http://ardx.org/VIDE06>



**CODE** (no need to type everything in just click)**Download the Code from ( <http://ardx.org/CODE06> )**

(copy the text and paste it into an empty Arduino Sketch)

```

/* Melody
 * (c) 2005 D. Cuartielles for K3
 *
 * This example uses a piezo speaker to play melodies. It sends
 * a square wave of the appropriate frequency to the piezo,
 * generating the corresponding tone.
 *
 * The calculation of the tones is made following the
 * mathematical operation:
 *
 *      timeHigh = period / 2 = 1 / (2 * toneFrequency)
 *
 * where the different tones are described as in the table:
 *
 * note    frequency period    timeHigh
 * C        261 Hz      3830    1915
 * d        294 Hz      3400    1700
 * e        329 Hz      3038    1519
 * f        349 Hz      2864    1432
 * g        392 Hz      2550    1275
 * a        440 Hz      2272    1136
 * b        493 Hz      2028    1014
 * C        523 Hz      1912    956
 *
 * http://www.arduino.cc/en/Tutorial/Melody
 */
int speakerPin = 9;
int length = 15; // the number of notes
char notes[] = "ccggaagffeeddc "; // a space represents a rest
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 4 };
int tempo = 300;

void playTone(int tone, int duration) {
  for (long i = 0; i < duration * 1000L; i += tone * 2) {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(speakerPin,
LOW);
    delayMicroseconds(tone);
  }
}

void playNote(char note, int duration) {
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'c' };
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };
  // play the tone corresponding to the note name
  for (int i = 0; i < 8; i++) {
    if (names[i] == note) {
      playTone(tones[i], duration);
    }
  }
}

void setup() {
  pinMode(speakerPin, OUTPUT);
}

void loop() {
  for (int i = 0; i < length; i++) {
    if (notes[i] == ' ') {
      delay(beats[i] * tempo); // rest
    } else {
      playNote(notes[i], beats[i] * tempo);
    }
    // pause between notes
    delay(tempo / 2);
  }
}

```

**NOT WORKING?** (3 things to try)**No Sound**

Given the size and shape of the piezo element it is easy to miss the right holes on the breadboard. Try double checking its placement.

**Can't Think While the Melody is Playing?**

Just pull up the piezo element whilst you think, upload your program then plug it back in.

**Tired of Twinkle Twinkle Little Star?**

The code is written so you can easily add your own songs, check out the code below to get started.

**MAKING IT BETTER****Playing with the speed:**

The timing for each note is calculated based on variables, as such we can tweak the sound of each note or the timing. To change the speed of the melody you need to change only one line.

```
int tempo = 300; --> int tempo = (new #)
```

Change it to a larger number to slow the melody down, or a smaller number to speed it up.

**Tuning the notes:**

If you are worried about the notes being a little out of tune this can be fixed as well. The notes have been calculated based on a formula in the comment block at the top of the program. But to tune individual notes just adjust their values in the tones[] array up or down until they sound right. (each note is matched by its name in the names[] (array ie. c = 1915)

```
char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };
```

**Composing your own melodies:**

The program is pre-set to play 'Twinkle Twinkle Little Star' however the way it is programmed makes changing the song easy. Each song is defined in one int and two arrays, the int length defines the number of notes, the first array notes[] defines each note, and the second beats[] defines how long each note is played. Some Examples:

Twinkle Twinkle Little Star

```
int length = 15;
char notes[] = {"ccggaagffeeddc "};
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 4 };
```

Happy Birthday (first line)

```
int length = 13;
char notes[] = {"ccdcfecdcgcf "};
int beats[] = {1,1,1,1,1,2,1,1,1,1,1,2,4};
```

**MORE, MORE, MORE:**

More details, where to buy more parts, where to ask more questions:

**<http://ardx.org/CIRC06>**